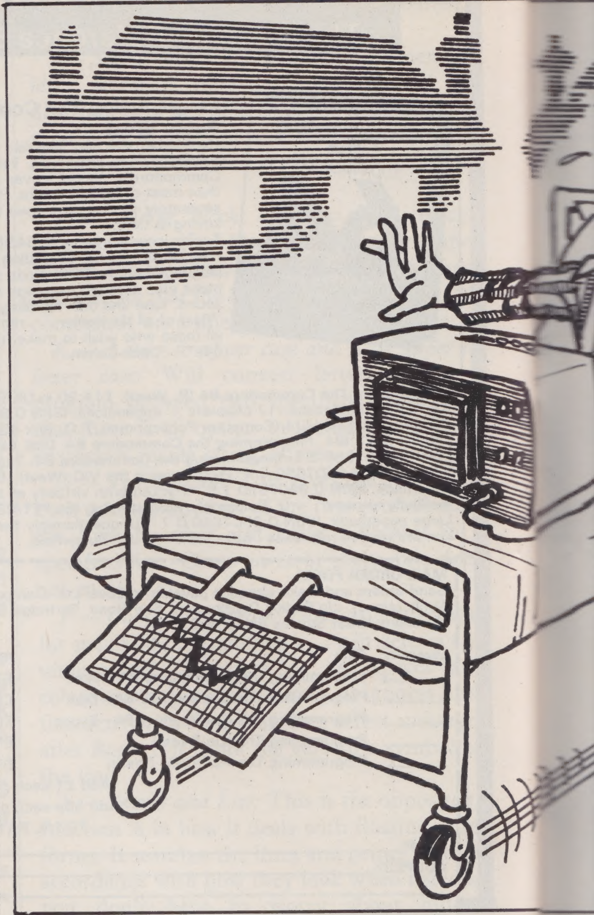


COMP

THE AIM OF this and the next article is to describe the construction of a compiler system for BBC Basic that is suitable for use on the BBC and Electron computers. The limitations on the amount of memory available on these machines mean that it is not possible for the program to compile all of BBC Basic — instead only a subset is treated but this is more extensive than that handled by some commercially available compilers. This subset is shown in figure 1. I shall refer to it as TC — Tiny Compiled — Basic.

TC Basic consists of two programs; the compiler which is a large Basic program and the run time system — RTS — which is a hybrid machine code — Basic program. This article will describe the compiler while the subsequent one will contain a description of the assembly language program that generates the RTS: this is called the run time system generator — RTSG.

The aim of a compiler, is to convert a program written in a high-level language like BBC Basic into a much simpler one like machine code which will therefore run faster. A program compiled by the TC Basic compiler will run as fast as a Forth program — ie., four to five times faster than the usual interpreted BBC Basic.

The code version of a program is rather more compact than the original Basic program and, typically, at least 10K of code can be generated on a BBC micro — for the Electron 4K would be a reasonable estimate.

The compilers usually implemented on mainframe and minicomputers take the high-level program and convert it into machine code. However, for the BBC computer this approach has the disadvantage that the 6502 CPU chip does not have instructions to do many of the things one would like it to.

For instance, there are no instructions to add two 16 bit numbers or for multiplication. To get round this I have invented a "virtual machine". This is an idealised CPU which has instructions to do all the things required. It is implemented as a 6502 machine-code program. The compiler produces machine-code for this virtual machine; such code is usually called pseudo-code or more briefly, p-code.

The machine-code to interpret the p-code, is contained in the RTS and this is, of course, the main purpose of this program. The use of p-code has some advantages and some disadvantages. The chief advantage is that it makes the compiler and run time system a bit less complex. A further advantage is that compiled programs are very compact. The chief disadvantage is that p-code executes about three times slower than machine code.

An alternative to p-code would be to compile the Basic program into 6502 machine-code statements. Every time some feature not in the 6502 instruction set was required we could jump to a subroutine in a run time support system. Simple functions like addition could be implemented as in line code. The major advantage of this approach is that programs run faster. However, there is the disadvantage that programs are very long; perhaps three to four times the length of an equivalent p-code program.

All the components of a p-code system are used by a true machine-code compiler. It is therefore worth starting with such a system if you want to learn how to make a compiler work.

The TC Basic compiler is shown in the listing. Obviously, if you want a TC Basic system, begin by typing in this listing. It should be Saved before you try Running it. In addition, before you can declare the compiler finished, it is necessary to enter in line 120 the value of the address in memory of the top of the RTS. This number, is provided by the RTSG program to be described in part 2.

The following is a desciption of the major components of the compiler. The main program runs between lines 150 and 300. Line 150 calls Procsetup — line 560. This initialises the values of various variables; in particular, it gives a large number of them the values of the Basic tokens which correspond to the keywords the compiler needs to know about. Procsetup also calls Proctok — lines 380 to 540 — which assembles a small machine-code program which is used to print out keywords from their tokens in the listings of programs. In addition, Procsetup prompts the user to specify the sizes of various things and reserves the appropriate quantities of memory.

Line 160 is a call of Procopen — lines 940 to 1020 — which opens the file containing the program to be compiled. Once the file is open, characters are always read from it by using FNCH — line 1020. Line 170 set things up so that compilation can start and in line 180, the call of Procprog actually initiates the compilation process. After the program has been compiled, execution of the main program continues at line 190. Lines 190 to 210 check if the storage allocations have been exceeded and print error messages if this is the case.

At line 220, a check is made to see if there have been any errors so far. If there have, the program branches to line 290: this is where the compiler ends if there are errors; a message is printed out giving the number of errors.

At this point, the compiler has constructed a p-code version of the Basic program. However, there are no actual addresses in this for either variables or for Goto — or in p-code jump — statements to branch to. To complete the p-code program it must be "fixed up". This is done by Procfix — lines 4050 to 4230 — and its associated routines in lines 3950 to 4280. This procedure allocates memory to variables and finds out the addresses in code to which Gotos, procedures and functions correspond. While the code is being fixed up, it is also relocated so that it will run at the address corresponding to the top of the RTS. Hence the reason why the value of this must

Figure 1. TC Basic Specification

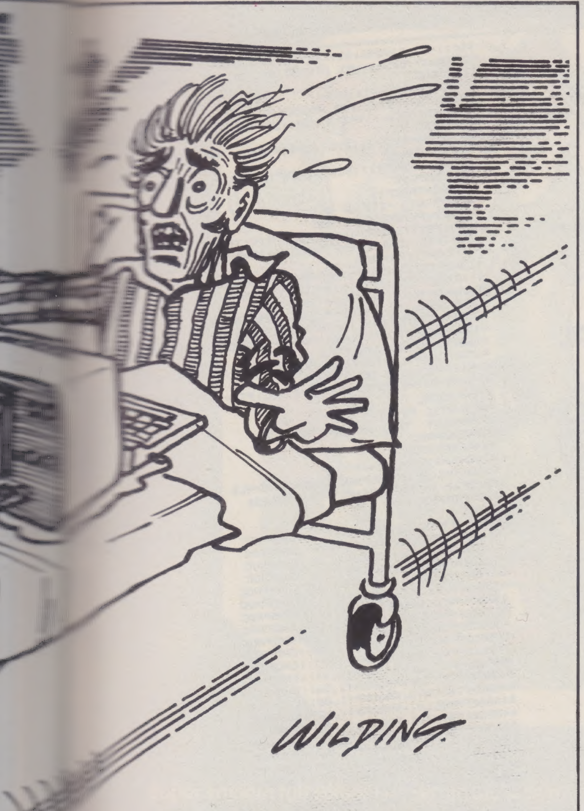
Variables:
2 byte signed integers.
Variable names may contain up to 255 characters and may end with a %. The variables A% to Z%, use the same memory locations as the usual Basic system variables thus allowing programs to pass information, and enabling parameter passing to the Call statement.
Arrays:
Multiple dimension arrays of 2 byte signed integers.
Arithmetic Operators: +, −, *, /, DIV, MOD
Logical Operators: AND, OR, EOR, NOT
Relational Operators: =, <, >, >=, <=, <>
Indirection Operator: ? (right hand side operand only)
Statements:
FOR TO STEP NEXT (no variable allowed after NEXT)
REPEAT UNTIL
PRINT"string" ' variable / expression, TAB(X,Y) ;
IF THEN ELSE
VDU , ;
* string (send string to OSCLI)
GOTO line number
SOUND ENVELOPE
MODE GCOL COLOUR CLS CLG MOVE DRAW PLOT
CALL END
DIM array / vairable
System variables / functions:

INKEY	GET	TRUE	FALSE	
ADVAL	TIME	ABS	SGN	
POS	VPOS	POINT	SQR	RND()

Procedures / functions:

DEFPROC	PROC	ENDPROC
DEFFN	FN	=

procedure and function arguments are not supported but procedures and functions can be used recursively.
In addition, the following are supported: REM, () (brackets), = (asignment), : (for multi-statement lines), integer constants, + and − (unary sign operators), & (for hex constants).



D J Piling with a compiler for the BBC.

PILER

be known.

Finally, if no errors occur in the fix up process the compiler ends at line 260; this calls Procdump — lines 1040 to 1180 — to save the code produced.

Some other features of the compiler are worth describing. The procedures in lines 1230 to 1900 compose the lexical analyser. Every time Proclx — lines 1230 to 1360 — is called, it reads the text token from the program and puts its type in the variable T%. Here "token" does not refer to the usual BBC Basic keyword token. Instead, it means the next significant piece of the program. Therefore, tokens can be variables and strings.

As was stated earlier, in a p-code system one constructs a virtual machine. In TC Basic the virtual machine is stack-based. This means that all operations take place either to or from the stack. For instance, there is an operation Add which takes a variable and adds it to the top element of the stack the result of the addition becoming the new top of stack element.

When the syntax analysis requires that a variable be added to the stack it calls Procadd. This writes the opcode for Add (2) into the code space and follows it with the address of the variable. In fact, until fix up instead of the variables address its number in the symbol table is used.

Other code generating subroutines, are supplied for all the operations the virtual machine is required to perform.

A typical scenario for the use of the compiler would be as follows: First, the program is developed as usual using the TC Basic subset to BBC Basic with the BBC Basic interpreter and Saved. Next, the compiler is Loaded and Run. It then asks for three pieces of information; the number of variables, the number of line numbers and the code space in bytes for the program to be compiled. Typically, answers of 100, 100 and 2000 would suffice for a small program. It is important to note that the number after a Goto counts as a variable as do the names of functions and procedures. Thus you must be careful when calculating how many variables to reserve space for.

Next, the compiler will ask you for the name of the program to be compiled. This will then be read in off the tape. This can in fact lead to a slight problem since, the compiler, is treating a program file as a data file. On a cassette-based system they are written to tape with different inter-block gaps.

This article is available on the Telsoft service.

With some cassette recorders having the wrong gap will lead to filing system errors when the program is being read in. The solution to this problem; if it happens to you, is to define one of the function keys as follows:

`*KEY0 *OPT,1 IIM INPUT '' "FILE NAME "A$:X% = OPENOUTA$: FORI% = PAGE TO TOP-1: BPUT#X%, ?I%: NEXT: CLOSE#X%: *OPT IIM`

f0 is then used for saving programs to be compiled.

As the program is compiled, it will be listed out on the screen. If any errors are found there will be a break in the listing and an appropriate error message. If enough room has not been reserved for variables code or lines you will be informed of the amount of space needed for the.

Finally, if there are no errors in compilation, a copy of the code equivalent of the Basic program can be saved. This will automatically be given the name C.name-of-compiled-program.

To execute the program, this new code file must be loaded into the RTS. The construction of this will be described in part 2. A C30 containing the listings from both articles, TC Basic systems set up for both tape and disc based computers and example programs and code files is available from D. J. Pilling, 12 Cherrywood Avenue, Anchorsholme, Blackpool. FY5 1SU. in return for a cheque for £2.99.

```
   10REM ================================
   20REM
   30REM TC BASIC COMPILER
   40REM D. J. Pilling
   50REM (c) January 1985
   60REM
   70REM ================================
   80
   90REM enter value for top of RTS in
  100REM line 120.
  110
  120toprts=7040
  130
  140MODE7:PROCT
  150PROCSETUP
  160PROCOPEN
  170T%=FNCH:A%=FNCH:PROCLX
  180PROCprog:CLOSE£Z%
  190IFNL%>ML% PROCEM(5)
  200IFNS%>MS% PROCEM(4)
  210IFE%<C%+MC% PROCEM(6)
  220IFEN% GOTO290
  230PROCFIX
  240IFEN% GOTO290
  250
  260PROCNE:PROCDUMP
  270END
  280
  290PROCEE
  300END
  310
  320DEFFNL(B1%,B2%,B3%)
  330B3%=B3%*256+B2%-&4040
  340IFB1%MOD&10=0 B3%=B3%+16384:B1%=B1%+4
  350IFB1%=&44 B1%=64 ELSEIFB1%=&54 B1%=0 ELSEIFB1%=&6
4 B1%=192 ELSEIFB1%=&74 B1%=128
  360=B1%+B3%
  370
  380DEFPROCTOK
  390IF?&8015=50B%=&7FF1 ELSEB%=&7FED
  400G%=&82ED:DIMTK 100
  410FORI%=0TO3STEP2:P%=TK:[OPTI%
  420CMP£128:BCCK%:BEQ L%
  430TAX:LDA£B%DIV256:STA&71:LDA£B%MOD256:STA&70:LDY£1
28
  440.D% TXA:CMP(&70),Y:BEQ C%
  450INC &70:BNE D%:INC &71
  460LDA&71:CMP G%DIV256:BNE D%
  470LDA&70:CMP G%MOD256:BNE D%:BEQ F%
  480.C% LDX£0
  490.H% DEY:INX:LDA(&70),Y:CMP£127:BCCH%
  500INY:INY:DEX:DEX:TXA:CMP£9:BCS F%
  510.E% LDA(&70),Y:JSR &FFEE:INY:DEX:BNE E%
  520.F% RTS:.K% JSR &FFEE:RTS
  530.L% LDA£65:JSR &FFEE:LDA£78:JSR &FFEE:LDA£68:JSR
&FFEE:RTS:]:NEXT
  540ENDPROC
  550
  560DEFPROCSETUP
  570PROCTOK
  580PRINT"SELECT STORAGE ALLOCATIONS"
  590INPUT'"ENTER MAX. NUMBER OF LINES"'"NOT LESS THAN
 10 >"ML%
  600IFML%<10GOTO590
  610INPUT'"ENTER NUMBER OF VARIABLES"'"BETWEEN 5 AND
256 >"MS%
  620IF MS%>256 OR MS%<5 GOTO610
  630INPUT'"ENTER SIZE OF CODE SPACE IN BYTES"'"NOT LE
SS THAN 100 >"MC%
  640IF MC%<100 GOTO630
  650DIM S$(MS%),S% 3*MS%,L% 4*ML%
  660S$=STRING$(40," "):S$=""
  670NL%=0:NS%=0:SN%=0:EN%=0:EL%=0
  680DIM C% MC%:C%=C%+4
  690RL%=toprts-C%+4:MC%=MC%-4
  700P%=C%:E%=C%+MC%
  710for=&E3:to=&B8:step=&88:next=&ED
  720vdu=&EF:print=&F1:tab=&8A:proc=&F2
  730def=&DD:endproc=&E1:rem=&F4
  740goto=&E5:end=&E0:inkey=&A6:if=&E7
  750else=&8B:lino=&8D:then=&8C:dim=&DE
  760eq=&3D:geq=&1:leq=&2:gt=&3E:lt=&3C
  770neq=&3:lb=&28:rb=&29:tr=&B9:fl=&A3
  780plus=&2B:minus=&2D:slash=&2F
  790times=&2A:pling=&3F:squote=&27
  800dquote=&22:comma=&2C:scol=&3B
  810col=&3A:spc=32:eoln=&D:eop=&FF
  820geq=&1:leq=&2:neq=&3:id=&4:lbl=&5
  830const=&6:string=&7:rep=&F5:unt=&FD
  840mod=&83:div=&81:sound=&D4:ar=&8
  850and=&80:or=&84:eor=&82:fn=&A4
  860timel=&D1:timer=&91:pc=37:ap=38
  870mode=&EB:cls=&DB:clg=&DA:gcol=&E6
  880colo=&FB:draw=&DF:move=&EC:plot=&F0
  890rnd=&B3:sqr=&B6:sgn=&B4:abs=&94
  900get=&A5:adv=&96:pos=&B1:vpo=&BC
  910poi=&B0:call=&D6:not=&AC:env=&E2
  920ENDPROC
  930
  940DEFPROCOPEN
  950*OPT1,1
  960*OPT2,1
  970PRINT''"ENTER NAME OF PROGRAM TO"
  980INPUT"COMPILE >"F$
  990Z%=OPENUPF$
 1000PRINT''"START OF COMPILATION"''
 1010*OPT1,0
 1020ENDPROC
 1030
 1040DEFPROCDUMP
 1050IFLENF$>8 F$=LEFT$(F$,8)
 1060F$="C."+F$
 1070PRINT''"READY TO SAVE CODE";F$
 1080PRINT"SAVE CODE ? Y/N ";
 1090I%=GET:IF I%=ASC"Y" ELSEIF I%=ASC"N"GOTO1170 ELSE
1090
 1100PRINT''
 1110*OPT1,1
 1120X%=OPENOUTF$
 1130PRINT£X%,C%,F%,E%,F$
 1140FOR I%=C%-4 TO E%:BPUT£X%,?I%:NEXT
 1150CLOSE£X%
 1160GOTO1070
 1170*OPT
 1180PRINT'':ENDPROC
 1190
 1200DEFFNCH:=BGET£Z%
 1210DEFPROCA:CALL TK:A%=FNCH:ENDPROC
 1220
 1230DEFPROCLX
 1240IFEL% T%=0:ENDPROC
 1250IFT%=eop:ENDPROC
 1260IFT%=eoln PROCNL
 1270IFA%=spc:REPEAT PROCA:UNTILA%<>spc
 1280IFNOTFNAL(A%) GOTO1320
 1290PROCIG:IFA%=pc S$=S$+"%":PROCA
 1300IFA%=lb T%=ar ELSET%=id
 1310PROCS:ENDPROC
 1320IFA%<>proc ANDA%<>fn GOTO1350
 1330T%=A%:PROCA:IFNOTFNAL(A%) T%=0:ENDPROC
 1340PROCIG:PROCS:ENDPROC
 1350IF FNDN(A%) PROCDNG ELSEIF A%=ap PROCA:PROCHNG EL
SEIF A%=rem PROCRG ELSEIF A%=dquote PROCSG ELSEIF A%=l
ino PROCLG ELSEPROCTG
 1360ENDPROC
 1370DEFFNAL(A%):=(A%>64ANDA%<91)OR(A%>96ANDA%<123)
 1380DEFFNDN(A%):=(A%>47ANDA%<58)
 1390DEFFNHN(A%):=(A%>47ANDA%<58)OR(A%>64ANDA%<71)
 1400DEFPROCLG
```

(listing continued on next page)

(listing continued from previous page)

```
1410N%=FNL(FNCH,FNCH,FNCH)
1420T%=lino:A%=FNCH:S$=STR$N%
1430PROCS
1440PRINT;N%;
1450ENDPROC
1460DEFPROCIG
1470S$="":REPEAT:S$=S$+CHR$A%:PROCA:UNTIL NOT(FNDN(A%
)ORFNAL(A%))
1480ENDPROC
1490DEFPROCDNG
1500N%=0:REPEAT N%=N%*10:N%=N%+A%-ASC"0":PROCA:UNTILN
OTFNDN(A%)
1510T%=const
1520ENDPROC
1530DEFPROCHNG
1540N%=0:REPEAT N%=N%*16
1550IFFNDN(A%) N%=N%+A%-ASC"0":PROCA ELSEIFFNHN(A%) N
%=N%+A%-ASC"A"+10:PROCA
1560UNTILNOTFNHN(A%)
1570T%=const
1580ENDPROC
1590DEFPROCRG
1600REPEAT PROCA:UNTILA%=eoln:T%=A%:PROCA
1610ENDPROC
1620DEFPROCSG
1630S$="":PROCA
1640REPEAT:S$=S$+CHR$A%:VDUA%:A%=FNCH:UNTIL A%=dquote
ORA%=eoln:IFA%=eoln PROCEM(1) ELSE PROCA
1650IFA%=dquote GOTO1640
1660T%=string
1670ENDPROC
1680DEFPROCCLIG
1690S$=CHR$A%
1700REPEAT:PROCA:S$=S$+CHR$A%:UNTIL A%=eoln:T%=A%:PRO
CA
1710ENDPROC
1720DEFPROCTG
1730T%=A%:PROCA
1740IFT%=gt ANDA%=eq T%=geq:PROCA
1750IFT%=lt ANDA%=eq T%=leq:PROCA
1760IFT%=lt ANDA%=gt T%=neq:PROCA
1770IFT%=col ANDA%=col REPEATPROCA:UNTIL(A%<>col)AND(
A%<>spc)
1780IFT%=col ANDA%=eoln:T%=A%:PROCA
1790ENDPROC
1800DEFPROCNL
1810IFA%=eop:ENDPROC
1820LI%=A%*256+FNCH:LL%=FNCH
1830IFNL%>ML% EN%=EN%+1:GOTO1880
1840L%?(NL%*4)=LI%DIV256
1850L%?(NL%*4+1)=LI%MOD256
1860L%?(NL%*4+2)=P%DIV256
1870L%?(NL%*4+3)=P%MOD256
1880NL%=NL%+1
1890PRINT';LI%; TAB(4);:A%=32
1900ENDPROC
1910
1920DEFPROCS
1930IF NS%>MS% EN%=EN%+1:ENDPROC
1940SN%=-1:REPEAT SN%=SN%+1
1950UNTIL (S$(SN%)=S$ AND S%?(3*SN%)=T%) OR S$(SN%)="
"
1960IFS$(SN%)="" S$(SN%)=S$:NS%=NS%+1:S%?(3*SN%)=T%:S
%?(3*SN%+1)=0:S%?(3*SN%+2)=0
1970ENDPROC
1980
1990DEFPROCprog
2000IFT%=eop PROCEND:ENDPROC
2010PROCstate:IFT%=col ORT%=eoln PROCLX:GOTO2000
2020ENDPROC
2030DEFPROCdef:PROCEND
2040PROCLX:IFT%<>proc ANDT%<>fn PROCC(proc)
2050S%?(3*SN%+1)=P%DIV256
2060S%?(3*SN%+2)=P%MOD256
2070IFT%=fn PROCLX:GOTO2120
2080PROCLX
2090IFT%=eop ORT%=endproc GOTO2110
2100PROCstate:IFT%=col ORT%=eoln PROCLX:GOTO2090
2110PROCC(endproc):PROCLX:PROCRTS:ENDPROC
2120IFT%=eop ORT%=eq GOTO2140
2130PROCstate:IFT%=col ORT%=eoln PROCLX:GOTO2120
2140PROCC(eq):PROCLX:PROCexp:PROCRTF:ENDPROC
2150
2160DEFPROCstate
2170IFT%=eoln ORT%=col ORT%=else:GOTO2450
2180IFT%=id PROCasign:GOTO2450
2190IFT%=ar PROCarasign:GOTO2450
2200IFT%=pling PROCaasign:GOTO2450
2210IFT%=timel PROCtasign:GOTO2450
2220IFT%=proc PROCproc:GOTO2450
2230IFT%=rep PROCrep:GOTO2450
2240IFT%=vdu PROCvdu:GOTO2450
2250IFT%=if PROCif:GOTO2450
2260IFT%=for PROCfor:GOTO2450
2270IFT%=goto PROCgoto:GOTO2450
2280IFT%=print PROCprint:GOTO2450
2290IFT%=end PROCend:GOTO2450
2300IFT%=def PROCdef:GOTO2450
2310IFT%=sound PROCsound:GOTO2450
2320IFT%=env PROCenv:GOTO2450
2330IFT%=call PROCcall:GOTO2450
2340IFT%=dim PROCdim:GOTO2450
2350IFT%=times PROCCLIG:PROCCLI:GOTO2450
2360IFT%=mode PROCmode:GOTO2450
2370IFT%=colo PROCcolo:GOTO2450
2380IFT%=gcol PROCgcol:GOTO2450
2390IFT%=move PROCmove:GOTO2450
2400IFT%=draw PROCdraw:GOTO2450
2410IFT%=plot PROCplot:GOTO2450
2420IFT%=cls PROCLX:PROCVDL(12):GOTO2450
2430IFT%=clg PROCLX:PROCVDL(16):GOTO2450
2440PROCEM(7):PROCELP
2450IFEL% PROCELP
2460IF(T%<>eoln ANDT%<>col ANDT%<>else) PROCEM(7):PRO
CELP
2470ENDPROC
2480
2490DEFPROCend:PROCEND:PROCLX:ENDPROC
2500DEFPROCasign:LOCALST%:ST%=SN%
2510PROCLX:PROCC(eq):PROCLX:PROCexp
2520PROCSTA:PROCADR(P%-3,ST%):ENDPROC
2530DEFPROCarasign:LOCALST%:ST%=SN%
2540PROCgma:PROCC(eq):PROCLX:PROCexp
2550PROCSTM:PROCADR(P%-3,ST%):ENDPROC
2560DEFPROCaasign:PROCLX:PROCsexp:PROCC(eq):PROCLX:PR
OCexp:PROCSTI:ENDPROC
2570DEFPROCtasign:PROCLX:PROCC(eq):PROCLX:PROCexp:PRO
CSTT:ENDPROC
2580DEFPROCproc:PROCJSR:PROCADR(P%-3,SN%):PROCLX:ENDP
ROC
2590DEFPROCfn:PROCJSR:PROCADR(P%-3,SN%):PROCLX:ENDPRO
C
2600DEFPROCvdu
2610PROCLX
2620PROCexp:IFT%=scol PROCVDD ELSEPROCVDU
2630IFT%=comma GOTO2610 ELSEIFT%=scol PROCLX:IFFNEXP
GOTO2620
2640ENDPROC
2650DEFPROCif:LOCAL L1%,L2%
2660PROCLX
2670PROCexp
2680L1%=P%:PROCJMC
2690IFT%=then PROCLX:IFT%=lino PROCJMP:PROCADR(P%-3,S
N%):PROCLX:GOTO2710
2700PROCstate
2710IFT%=col:PROCLX:GOTO2700
2720IFT%<>else PROCADR(L1%,P%):ENDPROC
2730L2%=P%:PROCJMP:PROCADR(L1%,P%)
2740PROCLX
2750IFT%=lino PROCJMP:PROCADR(P%-3,SN%):PROCLX:GOTO27
70
2760PROCstate
2770IFT%=col:PROCLX:GOTO2760
2780PROCADR(L2%,P%)
2790ENDPROC
2800DEFPROCfor:LOCAL ST%:PROCLX
2810PROCC(id):ST%=SN%:PROCLX
2820PROCC(eq):PROCLX:PROCexp
2830PROCSTA:PROCADR(P%-3,ST%)
2840PROCC(to):PROCLX:PROCexp
2850IFT%=step PROCLX:PROCexp ELSEPROCLDL(1)
2860PROCLX:PROCFXR:PROCADR(P%-3,ST%)
2870IFT%=eop ORT%=next GOTO2890
2880PROCstate:IFT%=col ORT%=eoln PROCLX:GOTO2870
2890PROCC(next):PROCLX:PROCNXT:ENDPROC
2900DEFPROCgoto:PROCLX:PROCC(lino):PROCJMP:PROCADR(P%
-3,SN%):PROCLX:ENDPROC
2910DEFPROCprint:LOCALT1%
2920T1%=T%:PROCLX
2930IFT%=squote PROCVDL(13):PROCVDL(10):GOTO2920
2940IFT%=scol GOTO2920
2950IFT%=string PROCSTR:GOTO2920
2960IFT%=comma PROCVDL(32):GOTO2920
2970IFT%=tab T1%=T%:PROCVDL(31):PROCLX:PROCexp:PROCC(
comma):PROCLX:PROCVDU:PROCexp:PROCC(rb):PROCLX:PROCVDU
:GOTO2930
2980IFFNEXP T1%=id:PROCexp:PROCPRI:GOTO2930
2990IFT1%<>scol PROCVDL(13):PROCVDL(10)
3000ENDPROC
3010DEFFNEXP:=T%=plus ORT%=minus ORT%=id ORT%=lb ORT%
=pling ORT%=const ORT%=inkey ORT%=timer ORT%=tr ORT%=f
l ORT%=sqr ORT%=rnd ORT%=sgn ORT%=abs ORT%=get ORT%=ad
v ORT%=pos ORT%=vpo ORT%=poi ORT%=not ORT%=fn ORT%=ar
3020DEFPROCSTR
3030IFLENS$=1 PROCVDL(ASCS$) ELSE PROCVDS
3040ENDPROC
3050DEFPROCexp:PROClterm
3060IFT%=or PROCLX:PROClterm:PROCORA:GOTO3060 ELSEIFT
%=eor PROCLX:PROClterm:PROCEOR:GOTO3060
3070ENDPROC
3080DEFPROClterm
3090PROClfact
3100IFT%=and PROCLX:PROClfact:PROCAND:GOTO3100
3110ENDPROC
3120DEFPROClfact:LOCAL T1%:PROCsexp
3130IFT%<>gt ANDT%<>lt ANDT%<>geq ANDT%<>leq ANDT%<>e
q ANDT%<>neq GOTO3160
3140T1%=T%:PROCLX:PROCsexp
3150IFT1%=gt PROCGTT ELSEIFT1%=lt PROCLTT ELSEIFT1%=g
eq PROCGEQ ELSEIFT1%=leq PROCLEQ ELSEIFT1%=eq PROCEQU
ELSEIFT1%=neq PROCNEQ
3160ENDPROC
3170DEFPROCsexp
3180IFT%=plus PROCLX:PROCterm ELSEIFT%=minus PROCLX:P
ROCterm:PROCNEG ELSE PROCterm
3190IFT%=plus PROCLX:PROCterm:PROCADD:GOTO3190 ELSEIF
T%=minus PROCLX:PROCterm:PROCSUB:GOTO3190
3200ENDPROC
3210DEFPROCfactor:LOCAL ST%:ST%=SN%
3220IFT%=id PROCLX:PROCLDA:PROCADR(P%-3,ST%):ENDPROC
3230IFT%=ar PROCgma:PROCLDM:PROCADR(P%-3,ST%):ENDPROC
3240IFT%=const PROCLX:PROCLDL(N%):ENDPROC
3250IFT%=pling PROCLX:PROClerm:PROCLDI:ENDPROC
3260IFT%=timer PROCLX:PROCLDT:ENDPROC
3270IFT%=fn PROCfn:ENDPROC
3280IFT%=lb PROCLX:PROCexp:PROCLX:ENDPROC
3290IFT%=inkey PROCLX:PROClerm:PROCINK:ENDPROC
3300IFT%=sqr PROCLX:PROClerm:PROCSQR:ENDPROC
3310IFT%=sgn PROCLX:PROClerm:PROCSGN:ENDPROC
3320IFT%=abs PROCLX:PROClerm:PROCABS:ENDPROC
3330IFT%=adv PROCLX:PROClerm:PROCADV:ENDPROC
3340IFT%=poi PROCLX:PROCexp:PROCC(comma):PROCLX:PROCe
xp:PROCC(rb):PROCLX:PROCPOI:ENDPROC
3350IFT%=rnd PROCLX:PROCC(lb):PROCLX:PROCexp:PROCC(rb
):PROCLX:PROCRND:ENDPROC
3360IFT%=tr PROCLX:PROCLDL(&FFFF):ENDPROC
3370IFT%=get PROCLX:PROCGET:ENDPROC
3380IFT%=pos PROCLX:PROCPOS:ENDPROC
3390IFT%=vpo PROCLX:PROCVPO:ENDPROC
3400IFT%=fl PROCLX:PROCLDL(0):ENDPROC
3410IFT%=not PROCLX:PROClerm:PROCNOT:ENDPROC
3420PROCC(1)
3430ENDPROC
3440DEFPROCterm
3450PROCfactor
3460IFT%=times PROCLX:PROCfactor:PROCMUL:GOTO3460 ELS
EIFT%=slash PROCLX:PROCfactor:PROCDIV:GOTO3460
3470IFT%=div PROCLX:PROCfactor:PROCDIV:GOTO3460 ELSEI
FT%=mod PROCLX:PROCfactor:PROCMOD:GOTO3460
3480ENDPROC
3490DEFPROClerm
3500IFT%=plus PROCLX:PROCfactor ELSEIFT%=minus PROCLX
:PROCfactor:PROCNEG ELSE PROCfactor
3510ENDPROC
3520DEFPROCsound
3530PROCLX:PROCexp:PROCC(comma)
3540PROCLX:PROCexp:PROCC(comma)
3550PROCLX:PROCexp:PROCC(comma)
3560PROCLX:PROCexp:PROCSND:ENDPROC
3570DEFPROCenv
3580LOCALI%:FOR I%=1TO13
3590PROCLX:PROCexp:PROCC(comma):NEXT
3600PROCLX:PROCexp:PROCENV:ENDPROC
3610DEFPROCcall:PROCLX:PROCexp:PROCCAL:ENDPROC
3620DEFPROCdim:LOCAL ST%
3630PROCLX:IFT%<>ar GOTO3660
3640ST%=SN%:IF S%?(SN%*3+1)<>0 PROCEM(8):PROCELP:ENDP
ROC
3650PROCgma:PROCLDL(S%?(ST%*3+2)):PROCDMA:PROCADR(P%-
3,ST%):S%?(ST%*3+1)=1:GOTO3680
3660IFT%<>id ENDPROC
3670ST%=SN%:PROCLX:PROCexp:PROCDMM:PROCADR(P%-3,ST%)
3680IFT%=comma:GOTO3630
3690ENDPROC
3700DEFPROCgma:LOCAL I%
3710PROCLX:PROCLX:I%=S%?(ST%*3+2):IFI%=0:I%=1:GOTO374
0
3720REPEAT I%=I%-1:PROCexp:IFI%<>0 PROCC(comma):PROCL
X
3730UNTILI%=0:PROCC(rb):PROCLX:ENDPROC
3740PROCexp:IFT%=comma:I%=I%+1:PROCLX:GOTO3740 ELSEPR
OCC(rb):PROCLX:S%?(ST%*3+2)=I%:ENDPROC
3750DEFPROCmode:PROCLX:PROCexp:PROCVDL(22):PROCVDU:EN
DPROC
3760DEFPROCcolo:PROCLX:PROCexp:PROCVDL(17):PROCVDU:EN
DPROC
3770DEFPROCgcol:PROCLX:PROCVDL(18):PROCexp:PROCVDU:PR
OCC(comma):PROCLX:PROCexp:PROCVDU:ENDPROC
3780DEFPROCplot:PROCLX:PROCVDL(25):PROCexp:PROCVDU:PR
OCC(comma):PROCLX:PROCexp:PROCVDD:PROCC(comma):PROCLX:
PROCexp:PROCVDD:ENDPROC
3790DEFPROCmove:PROCLX:PROCVDL(25):PROCVDL(4):PROCexp
:PROCVDD:PROCC(comma):PROCLX:PROCexp:PROCVDD:ENDPROC
3800DEFPROCdraw:PROCLX:PROCVDL(25):PROCVDL(5):PROCexp
:PROCVDD:PROCC(comma):PROCLX:PROCexp:PROCVDD:ENDPROC
3810DEFPROCrep:LOCALL1%:PROCLX:L1%=P%
3820IFT%=eop ORT%=unt GOTO3840
3830PROCstate:IFT%=col ORT%=eoln PROCLX:GOTO3820
3840PROCC(unt):PROCLX:PROCexp
3850PROCJMC:PROCADR(P%-3,L1%):ENDPROC
3860DEFPROCLFIX(SX%)
3870LI%=VALS$(SN%)
3880I%=0:REPEAT
3890IFLI%=L%?(4*I%)*256+L%?(4*I%+1):S%?(SX%+1)=L%?(4*
I%+2):S%?(SX%+2)=L%?(4*I%+3):I%=NL%
3900I%=I%+1
3910UNTIL I%>=NL%
3920IFI%=NL% PROCEM(2)
3930ENDPROC
3940
3950DEFPROCIFIX(SX%)
3960IFLENS$(SN%)=2:IFRIGHT$(S$(SN%),1)="%":I%=ASCLEFT
$(S$(SN%),1):IF(I%>64ANDI%<91):I%=(I%-65)*4+&404:S%?(S
X%+1)=I% DIV256:S%?(SX%+2)=I% MOD256:ENDPROC
3970S%?(SX%+1)=P% DIV256:S%?(SX%+2)=P% MOD256:P%=P%+2
:ENDPROC
3980DEFPROCAFIX(SX%)
3990IFS%?(SX%+1)=0 PROCEM(9)
4000S%?(SX%+1)=P% DIV256:S%?(SX%+2)=P% MOD256:P%=P%+2
:ENDPROC
4010DEFPROCPFIX(SX%):IFS%?(SX%+1)=0 PROCEM(3)
4020ENDPROC
4030DEFPROCFFIX(SX%):IFS%?(SX%+1)=0 PROCEM(3)
4040ENDPROC
4050DEFPROCFIX
4060PRINT''"FIX UP STAGE"
4070E%=P%:SN%=0:REPEAT
4080IFS%?(3*SN%)=lino:PROCLFIX(3*SN%)
4090IFS%?(3*SN%)=id PROCIFIX(3*SN%) ELSEIFS%?(3*SN%)=
ar PROCAFIX(3*SN%)
4100IFS%?(3*SN%)=proc:PROCPFIX(3*SN%)
4110IFS%?(3*SN%)=fn:PROCFFIX(3*SN%)
4120SN%=SN%+1:UNTILS$(SN%)=""
4130F%=P%:C%!-4=(F%+RL%)*&10000+E%+RL%
4140P%=C%:REPEAT:I%=?P%:IFI%=48P%=P%+1
4150IF(I%<64ORI%>100) P%=P%+1:GOTO4210 ELSEP%=P%+3
4160IFI%=74GOTO4210
4170IFI%=90 ORI%=92:P%=P%+(P%?-2)-1:GOTO4210
4180IFP%?-1<>0 GOTO4200
4190N%=P%?-2:P%?-1=S%?(N%*3+1):P%?-2=S%?(N%*3+2)
4200PROCRL
4210UNTIL P%>=E%
4220PRINT''"FIX UP NOW COMPLETE"
4230ENDPROC
4240DEFPROCRL
4250N%=256*P%?-1+P%?-2
4260IFN%<&E00 ENDPROC ELSEN%=N%+RL%
4270P%?-1=N%DIV256:P%?-2=N%MOD256
4280ENDPROC
4290
4300DEFPROCADD:PROCPP(1,2):ENDPROC
4310DEFPROCSUB:PROCPP(1,4):ENDPROC
4320DEFPROCMUL:PROCPP(1,6):ENDPROC
4330DEFPROCDIV:PROCPP(1,8):ENDPROC
4340DEFPROCGTT:PROCPP(1,10):ENDPROC
4350DEFPROCLTT:PROCPP(1,12):ENDPROC
4360DEFPROCGEQ:PROCPP(1,14):ENDPROC
4370DEFPROCLEQ:PROCPP(1,16):ENDPROC
4380DEFPROCEQU:PROCPP(1,18):ENDPROC
4390DEFPROCNEQ:PROCPP(1,20):ENDPROC
4400DEFPROCNEG:PROCPP(1,22):ENDPROC
4410DEFPROCVDU:PROCPP(1,24):ENDPROC
4420DEFPROCINK:PROCPP(1,26):ENDPROC
4430DEFPROCRTS:PROCPP(1,28):ENDPROC
4440DEFPROCLDI:PROCPP(1,30):ENDPROC
4450DEFPROCSTI:PROCPP(1,32):ENDPROC
4460DEFPROCEND:PROCPP(1,34):ENDPROC
4470DEFPROCPRI:PROCPP(1,36):ENDPROC
4480DEFPROCMOD:PROCPP(1,44):ENDPROC
4490DEFPROCVDD:PROCPP(1,46):ENDPROC
4500DEFPROCPP(B%,C%)
4510IFFNC(B%) ?P%=C%:P%=P%+B%
4520ENDPROC
4530DEFPROCVDL(L%):IFFNC(3):?P%=48:P%?1=L%:P%=P%+2
4540ENDPROC
4550DEFPROCSND:PROCPP(1,50):ENDPROC
4560DEFPROCLDT:PROCPP(1,52):ENDPROC
4570DEFPROCSTT:PROCPP(1,54):ENDPROC
4580DEFPROCAND:PROCPP(1,56):ENDPROC
4590DEFPROCORA:PROCPP(1,58):ENDPROC
4600DEFPROCEOR:PROCPP(1,60):ENDPROC
4610DEFPROCRND:PROCPP(1,62):ENDPROC
4620DEFPROCJMP:PROCPP(3,64):ENDPROC
4630DEFPROCJSR:PROCPP(3,66):ENDPROC
4640DEFPROCJMC:PROCPP(3,68):ENDPROC
4650DEFPROCLDA:PROCPP(3,70):ENDPROC
4660DEFPROCSTA:PROCPP(3,72):ENDPROC
4670DEFPROCFXR:PROCPP(3,76):ENDPROC
4680DEFPROCLDM:PROCPP(3,80):ENDPROC
4690DEFPROCSTM:PROCPP(3,82):ENDPROC
4700DEFPROCDMM:PROCPP(3,84):ENDPROC
4710DEFPROCDMA:PROCPP(3,86):ENDPROC
4720DEFPROCLDL(L%)
4730IFFNC(3) ?P%=74:P%?1=L%MOD256:P%?2=L%DIV256:P%=P%
+3
4740ENDPROC
4750DEFPROCVDS
4760IFFNC(2+LENS$) ?P%=90:P%?1=LENS$:$(P%+2)=S$:P%=P%
+2+LENS$
4770ENDPROC
4780DEFPROCCLI
4790IFFNC(2+LENS$) ?P%=92:P%?1=LENS$:$(P%+2)=S$:P%=P%
+2+LENS$
4800ENDPROC
4810DEFPROCSQR:PROCPP(1,102):ENDPROC
4820DEFPROCSGN:PROCPP(1,104):ENDPROC
4830DEFPROCABS:PROCPP(1,106):ENDPROC
4840DEFPROCGET:PROCPP(1,108):ENDPROC
4850DEFPROCADV:PROCPP(1,110):ENDPROC
4860DEFPROCPOS:PROCPP(1,112):ENDPROC
4870DEFPROCVPO:PROCPP(1,114):ENDPROC
4880DEFPROCPOI:PROCPP(1,116):ENDPROC
4890DEFPROCCAL:PROCPP(1,118):ENDPROC
4900DEFPROCNOT:PROCPP(1,120):ENDPROC
4910DEFPROCENV:PROCPP(1,122):ENDPROC
4920DEFPROCRTF:PROCPP(1,126):ENDPROC
4930DEFPROCNXT:PROCPP(1,132):ENDPROC
4940DEFPROCADR(P%,A%)
4950IFP%<C% OR P%+2>E% ENDPROC
4960P%?2=A%DIV256
4970P%?1=A%MOD256
4980ENDPROC
4990
5000DEFFNC(B%)
5010IFP%+B%>=E% EN%=EN%+1:E%=E%-B%:=FALSE ELSE:=TRUE
5020DEFPROCERT:PRINT"<RETURN> TO CONTINUE ";:REPEAT U
NTIL INKEY-74:PRINT:ENDPROC
5030DEFPROCEM(I%)
5040EN%=EN%+1
5050IFI%=1 PRINT'"missing """
5060IFI%=2 PRINT'"no such line number "+S$(SN%)
5070IFI%=3 PRINT'"no such fn/proc "+S$(SN%)
5080IFI%=4 PRINT'"too many variables"
5090IFI%=5 PRINT'"program needs ";NL%;" lines"
5100IFI%=6 PRINT'"program needs ";2*MC%+C%-E%;" bytes
"
5110IFI%=7 PRINT'"syntax error !"
5120IFI%=8 PRINT'"array already dim'd"
5130IFI%=9 PRINT'"array "+S$(SN%)+" not dim'd"
5140PROCERT:ENDPROC
5150DEFPROCC(D%)
5160IF EL% ENDPROC
5170IF T%=D% ENDPROC
5180PRINT'"missing ";
5190EL%=T%:T%=A%:A%=D%
5200IFA%>32 ANDA%<>lino THEN CALL TK ELSEIFA%=id PRIN
T"id / variable";ELSEIFA%=ar PRINT"array dim"; ELSEIFA
%=1 PRINT"expression"; ELSEIFA%=lino PRINT"line number
";
5210A%=T%:EN%=EN%+1:PRINT:T%=0:PROCERT:ENDPROC
5220DEFPROCELP
5230IFEL%=eoln OR EL%=eop T%=EL%:EL%=FALSE:ENDPROC
5240EL%=FALSE:REPEAT PROCLX:UNTIL T%=eoln ORT%=eop
5250ENDPROC
5260DEFPROCEE
5270PRINT'"";EN%;" ERRORS - COMPILATION STOPPED"
5280ENDPROC
5290DEFPROCNE
5300PRINT'"COMPILATION COMPLETE"
5310ENDPROC
5320DEFPROCT
5330FORI%=0TO23
5340VDU129+(I%/2)MOD5,157,141,135
5350PRINT"<<<  T.C.  BASIC  COMPILER  >>>"
5360NEXT:I%=INKEY150:CLS
5370ENDPROC
```

■

```
10REM===============================
20REM  T.C. BASIC
30REM  RTS GENERATOR
40REM  (c) D.J. Pilling
50REM  January 1985
60REM===============================
70
80REM enter the usual value of page
90REM for your system in line 110
100
110page%=&E00
120
130PROCSETUP
140PROCASSEMBLE
150PROCJUMPTAB
160PROCFINISH
170END
180
190DEFPROCASSEMBLE
200FOR PASS=0TO3 STEP 3
210P%=RT
220[OPT PASS
230
240.START
250 PLA:STA RL:PLA:STA RH
260 TSX:STX SP
270 LDY#0:LDA (PL),Y:STA R1L
280 INY:LDA (PL),Y:STA R1H
290 INY:LDA (PL),Y:STA FVL
300 INY:LDA (PL),Y:STA FVH
310 LDY#0:STY FP
320.SVS LDA#0:STA (R1L),Y
330 LDA FVH:CMP R1H:BNE SVI
340 LDA FVL:CMP R1L:BEQ SVX
350.SVI INCR1L:BNE SVS:INCR1H:BNE SVS
360.SVX LDA#4
370
380.SR
390 CLC:ADC PL:STA PL:BCC SF
400 INC PH:BNE SF
410.SX INC PL:BNE SF:INC PH
420.SF LDY#0:LDA (PL),Y:TAX
430 LDA JT,X:STA JTL:INX
440 LDA JT,X:STA JTH
450 JMP(JTL)
460
470.FVL BRK:.FVH BRK
480
490.add
500 PLA:STA R1L:PLA:STA R1H
510 PLA:CLC:ADC R1L:TAY
520 PLA:ADC R1H:PHA:TYA:PHA
530 JMP SX
540
550.sub
560 PLA:STA R1L:PLA:STA R1H
570 PLA:SEC:SBC R1L:TAY
580 PLA:SBC R1H:PHA:TYA:PHA
590 JMP SX
600
610.mul
620 LDA#0:STA SG
630 PLA:STA R1L:PLA:STA R1H
640 BPL M2
650 LDA#0:SEC:SBC R1L:STA R1L
660 LDA#0:SBC R1H:STA R1H
670 LDA#1:STA SG
680.M2 PLA:STA R2L:PLA:STA R2H
690 BPL M3
700 LDA#0:SEC:SBC R2L:STA R2L
710 LDA#0:SBC R2H:STA R2H
720 LDA#1:EOR SG:STA SG
730.M3 LDA#0:STA R3L:STA R3H
740 LDX#16:LDA R1H:BEQ ML
750 LDY R2H:STA R2H:STY R1H
760 LDA R1L:LDY R2L:STA R2L:STY R1L
770.ML CLC:ROR R2H:ROR R2L
780 BCS MA
790.MC CLC:ROL R1L:ROL R1H
800 DEX:BNE ML
810 LDA SG:BEQ MX
820 LDA#0:SEC:SBC R3L:TAX
830 LDA#0:SBC R3H:PHA:TXA:PHA
840 JMP SX
850.MX LDA R3H:PHA:LDA R3L:PHA
860 JMP SX
870.MA LDA R1L:CLC:ADC R3L:STA R3L
880 LDA R1H:ADC R3H:STA R3H
890 JMP MC
900
910.div
920 LDA#0:STA SG
930 PLA:STA R3L:PLA:STA R3H:TAX
940 ORA R3L:BNE DK
950 LDA#88:JMP EXIT
960.DK TXA:BPL D2
970 LDA#0:SEC:SBC R3L:STA R3L
980 LDA#0:SBC R3H:STA R3H
990 LDA#1:STA SG
1000.D2 PLA:STA R2L:PLA:STA R2H
1010 BPL D3
1020 LDA#0:SEC:SBC R2L:STA R2L
1030 LDA#0:SBC R2H:STA R2H
1040 LDA#1:EOR SG:STA SG
1050.D3 JSR DM
1060 LDA SG:BEQ DX
1070 LDA#0:SEC:SBC R2L:TAX
1080 LDA#0:SBC R2H:PHA:TXA:PHA
1090 JMP SX
1100.DX LDA R2H:PHA:LDA R2L:PHA
1110 JMP SX
1120.DM
1130 LDA#0:STA R1L:STA R1H
1140 LDX#16:CLC
1150.DL
1160 ROL R2L:ROL R2H
1170 ROL R1L:ROL R1H
```

```
1180 LDA R3H:CMP R1H:BEQ DQ
1190 BCS D0:BCC D1
1200.DQ LDA R3L:CMP R1L:BEQ D1
1210 BCS D0:BCC D1
1220.D0 CLC:BCC DN
1230.D1
1240 LDA R1L:SEC:SBC R3L:STA R1L
1250 LDA R1H:SBC R3H:STA R1H:SEC
1260.DN DEX:BNE DL
1270 ROL R2L:ROL R2H:RTS
1280
1290.mod
1300 LDA#0:STA SG
1310 PLA:STA R3L:PLA:STA R3H:TAX
1320 ORA R3L:BNE OK
1330 LDA#88:JMP EXIT
1340.OK TXA:BPL O2
1350 LDA#0:SEC:SBC R3L:STA R3L
1360 LDA#0:SBC R3H:STA R3H
1370.O2 PLA:STA R2L:PLA:STA R2H
1380 BPL O3
1390 LDA#0:SEC:SBC R2L:STA R2L
1400 LDA#0:SBC R2H:STA R2H
1410 LDA#1:STA SG
1420.O3 JSR DM
1430 LDA SG:BEQ OX
1440 LDA#0:SEC:SBC R1L:TAX
1450 LDA#0:SBC R1H:PHA:TXA:PHA
1460 JMP SX
1470.OX LDA R1H:PHA:LDA R1L:PHA
1480 JMP SX
1490
1500.gtt
1510 PLA:STA R1L:PLA:STA R1H
1520 PLA:STA R2L:PLA:STA R2H
1530 JSR CP
1540 BEQ CF:BMI CF:BNE CT
1550
1560.ltt
1570 PLA:STA R1L:PLA:STA R1H
1580 PLA:STA R2L:PLA:STA R2H
1590 JSR CP
1600 BEQ CF:BPL CF:BNE CT
1610
1620.geq
1630 PLA:STA R1L:PLA:STA R1H
1640 PLA:STA R2L:PLA:STA R2H
1650 JSR CP
1660 BMI CF:BPL CT
1670
1680.leq
1690 PLA:STA R1L:PLA:STA R1H
1700 PLA:STA R2L:PLA:STA R2H
1710 JSR CP
1720 BPL CL:BEQ CF:BNE CT
1730.CL BNE CF:BEQ CT
1740
1750.equ
1760 PLA:STA R1L:PLA:STA R1H
1770 PLA:STA R2L:PLA:STA R2H
1780 JSR CP
1790 BMI CF:BNE CF:BEQ CT
1800
1810.neq
1820 PLA:STA R1L:PLA:STA R1H
1830 PLA:STA R2L:PLA:STA R2H
1840 JSR CP
1850 BEQ CF:BNE CT
1860
1870.CT LDA#&FF:PHA:PHA:JMP SX
1880.CF LDA#&00:PHA:PHA:JMP SX
1890.CP LDA R2L:CMP R1L:BEQ CQ
1900 LDA R2H:SBC R1H:ORA #1
1910 BVS CO:RTS
1920.CQ LDA R2H:SBC R1H:BVS CO:RTS
1930.CO EOR #&80:ORA #1:RTS
1940
1950.neg
1960 PLA:STA R1L:PLA:STA R1H
1970 LDA#0:SEC:SBC R1L:TAY
1980 LDA#0:SBC R1H:PHA:TYA:PHA
1990 JMP SX
2000
2010.vdu
2020 PLA:JSR OSWRITE:PLA
2030 JMP SX
2040
2050.ink
2060 PLA:TAX:PLA:TAY
2070 LDA#&81:JSR OSBYTE
2080 TYA:CMP#&FF:BEQ IT
2090 CMP#&1B:BEQ IE
2100.IR TYA:PHA:TXA:PHA
2110 JMP SX
2120.IE LDA#&7E:JSR OSBYTE
2130 LDX#0:LDY#0:JMP IR
2140.IT LDX#&FF:JMP IR
2150
2160.rts
2170 PLA:STA PL:PLA:STA PH
2180 JMP SF
2190
2200.ldi
2210 PLA:STA R1L:PLA:STA R1H
2220 LDA#0:PHA
2230 LDY#0:LDA (R1L),Y:PHA
2240 JMP SX
2250
2260.sti
2270 PLA:TAY:PLA:TAX
2280 PLA:STA R1L:PLA:STA R1H
2290 TYA:LDY#0:STA (R1L),Y
2300 LDA#0:LDY#1:STA (R1L),Y
2310 JMP SX
2320
2330.end
2340 LDA#44:JMP EXIT
2350
2360.pri
```

This article is available on the Telsoft service.

COM

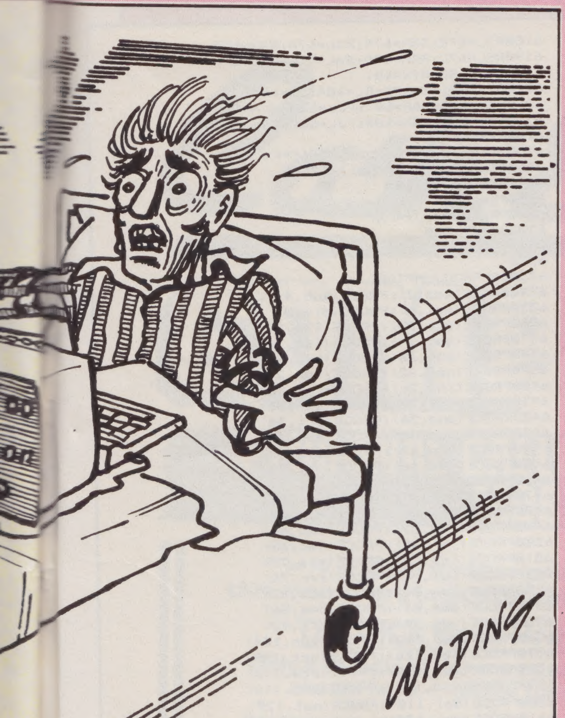
IN THIS ARTICLE, the TC Basic run time system generator — RTSG — will be described. This program generates the TC Basic run time system — RTS; the purpose of this, is to provide the facilities and functions required by TC Basic programs. In addition, the RTS must also interpret the p-code programs.

The RTS consists of a small Basic program followed by a substantial amount of machine code. To construct it, type in the listing of the RTSG in figure 1. Before doing this, type in

```
2370 PLA:STA R1L:STA R2L
2380 PLA:STA R1H:STA R2H
2390 BPL PB
2400 LDA #0:SEC
2410 SBC R1L:STA R1L:STA R2L
2420 LDA #0
2430 SBC R1H:STA R1H:STA R2H
2440 LDA #ASC"-":JSR OSWRITE
2450.PB
2460 LDA#0:STA SG
2470 LDA # 10000 MOD256:STA R3L
2480 LDA # 10000 DIV256:STA R3H
2490 JSR PS
2500 LDA # 1000 MOD256:STA R3L
2510 LDA # 1000 DIV256:STA R3H
2520 JSR PS
2530 LDA #100:STA R3L
2540 LDA #0:STA R3H
2550 JSR PS
2560 LDA #10:STA R3L
2570 JSR PS
2580 LDA #1:STA R3L:STA SG
2590 JSR PS
2600 JMP SX
2610.PS LDX#0
2620.PP LDA R1L:SEC:SBC R3L:STA R1L
2630 LDA R1H:SBC R3H:STA R1H
2640 BMI PE
2650 STA R2H:LDA R1L:STA R2L
2660 INX:JMP PP
2670.PE LDA R2L:STA R1L
2680 LDA R2H:STA R1H
2690 TXA:BNE PR:LDA SG:BEQ PX
2700.PR TXA:CLC:ADC #ASC"0"
```

```
360.SVX LDA#4
370
380.SR
390 CLC:ADC PL:STA PL:BCC SF
400 INC PH:BNE SF
410.SX INC PL:BNE SF:INC PH
420.SF LDY#0:LDA (PL),Y:TAX
430 LDA JT,X:STA JTL:INX
440 LDA JT,X:STA JTH
450 JMP(JTL)
460
470.FVL BRK:.FVH BRK
480
490.add
500 PLA:STA R1L:PLA:STA R1H
510 PLA:CLC:ADC R1L:TAY
520 PLA:ADC R1H:PHA:TYA:PHA
530 JMP SX
540
550.sub
560 PLA:STA R1L:PLA:STA R1H
570 PLA:SEC:SBC R1L:TAY
580 PLA:SBC R1H:PHA:TYA:PHA
590 JMP SX
600
610.mul
620 LDA#0:STA SG
630 PLA:STA R1L:PLA:STA R1H
640 BPL M2
650 LDA#0:SEC:SBC R1L:STA R1L
660 LDA#0:SBC R1H:STA R1H
670 LDA#1:STA SG
680.M2 PLA:STA R2L:PLA:STA R2H
690 BPL M3
700 LDA#0:SEC:SBC R2L:STA R2L
710 LDA#0:SBC R2H:STA R2H
720 LDA#1:EOR SG:STA SG
730.M3 LDA#0:STA R3L:STA R3H
740 LDX#16:LDA R1H:BEQ ML
750 LDY R2H:STA R2H:STY R1H
760 LDA R1L:LDY R2L:STA R2L:STY R1L
770.ML CLC:ROR R2H:ROR R2L
780 BCS MA
790.MC CLC:ROL R1L:ROL R1H
800 DEX:BNE ML
810 LDA SG:BEQ MX
820 LDA#0:SEC:SBC R3L:TAX
830 LDA#0:SBC R3H:PHA:TXA:PHA
840 JMP SX
850.MX LDA R3H:PHA:LDA R3L:PHA
860 JMP SX
870.MA LDA R1L:CLC:ADC R3L:STA R3L
880 LDA R1H:ADC R3H:STA R3H
890 JMP MC
900
910.div
920 LDA#0:STA SG
930 PLA:STA R3L:PLA:STA R3H:TAX
940 ORA R3L:BNE DK
950 LDA#88:JMP EXIT
960.DK TXA:BPL D2
970 LDA#0:SEC:SBC R3L:STA R3L
980 LDA#0:SBC R3H:STA R3H
990 LDA#1:STA SG
1000.D2 PLA:STA R2L:PLA:STA R2H
1010 BPL D3
1020 LDA#0:SEC:SBC R2L:STA R2L
1030 LDA#0:SBC R2H:STA R2H
1040 LDA#1:EOR SG:STA SG
1050.D3 JSR DM
1060 LDA SG:BEQ DX
1070 LDA#0:SEC:SBC R2L:TAX
1080 LDA#0:SBC R2H:PHA:TXA:PHA
1090 JMP SX
1100.DX LDA R2H:PHA:LDA R2L:PHA
1110 JMP SX
1120.DM
1130 LDA#0:STA R1L:STA R1H
1140 LDX#16:CLC
1150.DL
1160 ROL R2L:ROL R2H
1170 ROL R1L:ROL R1H
```

```
1540 BEQ CF:BMI CF:BNE
1550
1560.ltt
1570 PLA:STA R1L:PLA:STA R1H
1580 PLA:STA R2L:PLA:STA R2H
1590 JSR CP
1600 BEQ CF:BPL CF:BNE CT
1610
1620.geq
1630 PLA:STA R1L:PLA:STA R1H
1640 PLA:STA R2L:PLA:STA R2H
1650 JSR CP
1660 BMI CF:BPL CT
1670
1680.leq
1690 PLA:STA R1L:PLA:STA R1H
1700 PLA:STA R2L:PLA:STA R2H
1710 JSR CP
1720 BPL CL:BEQ CF:BNE CT
1730.CL BNE CF:BEQ CT
1740
1750.equ
1760 PLA:STA R1L:PLA:STA R1H
1770 PLA:STA R2L:PLA:STA R2H
1780 JSR CP
1790 BMI CF:BNE CF:BEQ CT
1800
1810.neq
1820 PLA:STA R1L:PLA:STA R1H
1830 PLA:STA R2L:PLA:STA R2H
1840 JSR CP
1850 BEQ CF:BNE CT
1860
1870.CT LDA#&FF:PHA:PHA:JMP SX
1880.CF LDA#&00:PHA:PHA:JMP SX
1890.CP LDA R2L:CMP R1L:BEQ CQ
1900 LDA R2H:SBC R1H:ORA #1
1910 BVS CO:RTS
1920.CQ LDA R2H:SBC R1H:BVS CO:RTS
1930.CO EOR #&80:ORA #1:RTS
1940
1950.neg
1960 PLA:STA R1L:PLA:STA R1H
1970 LDA#0:SEC:SBC R1L:TAY
1980 LDA#0:SBC R1H:PHA:TYA:PHA
1990 JMP SX
2000
2010.vdu
2020 PLA:JSR OSWRITE:PLA
2030 JMP SX
2040
2050.ink
2060 PLA:TAX:PLA:TAY
2070 LDA#&81:JSR OSBYTE
2080 TYA:CMP#&FF:BEQ IT
2090 CMP#&1B:BEQ IE
2100.IR TYA:PHA:TXA:PHA
2110 JMP SX
2120.IE LDA#&7E:JSR OSBYTE
2130 LDX#0:LDY#0:JMP IR
2140.IT LDX#&FF:JMP IR
2150
2160.rts
2170 PLA:STA PL:PLA:STA PH
2180 JMP SF
2190
2200.ldi
2210 PLA:STA R1L:PLA:STA R1H
2220 LDA#0:PHA
2230 LDY#0:LDA (R1L),Y:PHA
2240 JMP SX
2250
2260.sti
2270 PLA:TAY:PLA:TAX
2280 PLA:STA R1L:PLA:STA R1H
2290 TYA:LDY#0:STA (R1L),Y
2300 LDA#0:LDY#1:STA (R1L),Y
2310 JMP SX
2320
2330.end
2340 LDA#44:JMP EXIT
2350
2360.pri
```



D J Piling continues his compiler for the BBC. If you're speed crazy this program is for you.

PILER

PAGE = PAGE + 5120

This is necessary because the RTS is constructed at the usual value of PAGE for your system. In fact, the value of PAGE used by your computer, must be entered in line 110. For a Tape based system, this will be &E00 and for a disc system &1900. The RTS is therefore constructed underneath the RTSG in memory.

When you have typed in all of the program, Save it and then go through the usual debugging process. When Run, the RTSG will produce an assembly language listing, this will be followed by a statement of the value of the address of the top of the RTS. Note this number down. Next reset Page to its usual value and type End followed by Run. If everything is OK you will be greeted by the RTS menu.

Three functions are performed by the RTS and can be selected by pressing the number keys 1, 2 or 3. These are to load a code file produced by the compiler into the RTS and form, a free standing module. Second, execute any such module and, thirdly, Save any such module. Initially, you must select option 3. This will allow you to Save your copy of the RTS.

Now reload the compiler into your computer and enter in line 120 the value for the top of the RTS which the RTSG provided you wish. The compiler is now complete and can be Saved.

In the last article, it was described how to compile a program. After following this prescription, you will be left with a file whose name starts with C. and which contains the p-code version of your program. The next thing to do, is to Load the RTS and Run it. Now select option 1 and type in the name of the code file when prompted.

This will be Loaded and a module formed which can be Saved with option 3. In the future whenever you want to execute the compiled version of your program, the module can be Loaded and Run and option 2 — execute code — selected. The module will be saved with a file name prefixed by M.

The RTS can be Escaped from and reRun. In addition, if your p-code program gets into an endless loop; Break plus Old will also work.

The TC Basic system, is thus now complete. Some points about its use should be noted. First, there is a limitation on the use of delimiters like Next and Endproc not present in BBC Basic; this is that for each Defproc there can be only one Endproc and for each FOR there can be only one Next. Thus statements like: IF A = 2 Endproc; will not compile.

This is done to protect the programmer; returning from a non-existent subroutine, would have disastrous results. Similar restrictions apply to functions and Repeat Until loops. Secondly, if a Def statement is encountered in the path of execution, it will be treated as an End statement.

The RTS implements code to deal with run time errors: attempts to divide by zero, to take square roots of negative numbers or dimension arrays twice will gracefully terminate execution with an error message and return control to the Basic section of the RTS.

As you can see from the listing of the RTSG the final few lines — lines 6970 to 7270 — are in fact the Basic section of the RTS; procedure Proccopy — lines 6750 to 6830 — is used to copy this Basic down to the usual value of Page. The two Stops in line 6960 are used as a tag for this copying process and must be typed exactly as shown. This piece of the program is just used to handle the RTS menu and the three functions that can be selected: i.e. loading, saving and executing the mc.

Interpret p-code

The machine code part of the RTS has to be able to interpret p-code; it does this by using the opcode number — you may have noticed from the code generation procedures in the compiler that all opcodes are multiples of two — as an index to select the start address of a routine from a jump table. An indirect jump to this address then results in the required function being performed.

Thus most of the assembly language consists of a large number of routines — each represents one opcode. In the listing they can be clearly seen by being split up by blank lines; in addition, each one starts with a label in lower case letters which corresponds to the names of the code generating procedure in the compiler.

Finally, the value of the top of the RTS is printed out so that it can be inserted in the compiler. For a system with a Page value of &E00 this number will be 7040 while, for a system with a Page value of &1900, it will be 9856.

A C30 containing the listings from both articles, TC Basic systems set up for both tape and disc based computers and example programs and code files is available from D J Pilling, 12 Cherrywood Avenue, Anchorsholme, Blackpool FY5 1SU in return for £2.99.

```
2710 JSR OSWRITE
2720 LDA#1:STA SG
2730.PX RTS
2740
2750.dup
2760 PLA:TAY:PLA:TAX
2770 PHA:TYA:PHA:TXA:PHA:TYA:PHA
2780 JMP SX
2790
2800.swp
2810 PLA:TAY:PLA:TAX
2820 PLA:STA R1L:PLA:STA R1H
2830 TXA:PHA:TYA:PHA
2840 LDA R1H:PHA:LDA R1L:PHA
2850 JMP SX
2860
2870.drp
2880 PLA:PLA:JMP SX
2890
2900.vdd
2910 PLA:JSR OSWRITE
2920 PLA:JSR OSWRITE
2930 JMP SX
2940
2950.vdl
2960 LDY#1:LDA (PL),Y
2970 JSR OSWRITE
2980 LDA#2:JMP SR
2990
3000.snd
3010 LDX# R1L:LDY #0
3020 PLA:STA R4L:PLA:STA R4H
3030 PLA:STA R3L:PLA:STA R3H
3040 PLA:STA R2L:PLA:STA R2H
3050 PLA:STA R1L:PLA:STA R1H
3060 LDA#7:JSR OSWORD:JMP SX
3070
3080.ldt
3090 LDX# R1L:LDY #0
3100 LDA#1:JSR OSWORD
3110 LDA R1H:PHA:LDA R1L:PHA
3120 JMP SX
3130
3140.stt
3150 LDX# R1L:LDY #0
3160 PLA:STA R1L:PLA:STA R1H
3170 LDA#0:STA R2L:STA R2H:STA R3L
3180 LDA#2:JSR OSWORD:JMP SX
3190
3200.and
3210 PLA:STA R1L:PLA:STA R1H
3220 PLA:TAX:PLA:AND R1H:PHA
3230 TXA:AND R1L:PHA:JMP SX
3240
3250.ora
3260 PLA:STA R1L:PLA:STA R1H
3270 PLA:TAX:PLA:ORA R1H:PHA
3280 TXA:ORA R1L:PHA:JMP SX
3290
3300.eor
3310 PLA:STA R1L:PLA:STA R1H
3320 PLA:TAX:PLA:EOR R1H:PHA
3330 TXA:EOR R1L:PHA:JMP SX
3340 .R1 RTS:.R2 RTS
3350
3360.rnd
3370 PLA:STA R3L:PLA:STA R3H
3380 BMI RX:ORA R3L:CMP#2:BCC RX
3390 LDY#14
3400.RLO LDA R2:AND#96:STA R1L
3410 ROL A
3420 EOR R1L:ROL A:ROL A
3430 ROL R1:ROL R2
3440 DEY:BNE RLO
3450 LDA R2:AND#127:STA R2H
3460 LDA R1:STA R2L:JSR DM
3470 LDA#1:CLC:ADC R1L:TAY:LDA#0
3480 ADC R1H:PHA:TYA:PHA:JMP SX
3490.RX LDA#1:STA R1:STA R2
3500 LDA#0:PHA:LDA#1:PHA:JMP SX
3510
3520.jsr
3530 LDA PL:CLC:ADC#3:TAY
3540 LDA PH:ADC#0
3550 PHA:TYA:PHA
3560
3570.jmp
3580 LDY#1:LDA (PL),Y:TAX
3590 LDY#2:LDA (PL),Y:STA PH
3600 TXA:STA PL
3610 JMP SF
3620
3630.jmc
3640 PLA:BNE C1
3650 PLA:BNE C2
3660 BEQ jmp
3670.C1 PLA
3680.C2 LDA#3:JMP SR
3690
3700.lda
```

(continued on next page)

(continued from previous page)

```
3710 LDY#1:LDA (PL),Y:STA R1L
3720 LDY#2:LDA (PL),Y:STA R1H
3730 LDY#1:LDA (R1L),Y:PHA
3740 LDY#0:LDA (R1L),Y:PHA
3750 LDA#3:JMP SR
3760
3770.sta
3780 LDY#1:LDA (PL),Y:STA R1L
3790 LDY#2:LDA (PL),Y:STA R1H
3800 LDY#0:PLA:STA (R1L),Y
3810 LDY#1:PLA:STA (R1L),Y
3820 LDA#3:JMP SR
3830
3840.ldl
3850 LDY#2:LDA (PL),Y:PHA
3860 LDY#1:LDA (PL),Y:PHA
3870 LDA#3:JMP SR
3880
3890.fxr
3900 LDX FP:LDY#7
3910.FXL LDA FAL,Y:STA&500,X
3920 INX:DEY:BPL FXL:STX FP
3930 LDY#1:LDA (PL),Y:STA FAL
3940 INY:LDA (PL),Y:STA FAH
3950 PLA:STA FSL:PLA:STA FSH
3960 PLA:STA FLL:PLA:STA FLH
3970 LDA#3:CLC:ADC PL:STA PL:STA FJL
3980 LDA#0:ADC PH:STA PH:STA FJH
3990 JMP SF
4000
4010.ldm
4020 JSR GTA:LDY#1:LDA (R3L),Y:PHA
4030 DEY:LDA (R3L),Y:PHA:LDA#3:JMP SR
4040
4050.stm
4060 PLA:STA R6L:PLA:STA R6H:JSR GTA
4070 LDY#1:LDA R6H:STA (R3L),Y
4080 DEY:LDA R6L:STA (R3L),Y
4090 LDA#3:JMP SR
4100
4110.dmm
4120 LDY#1:LDA (PL),Y:STA R1L
4130 INY:LDA (PL),Y:STA R1H
4140 DEY:LDA FVH:STA (R1L),Y
4150 DEY:LDA FVL:STA (R1L),Y
4160 PLA:CLC:ADC FVL:STA FVL
4170 PLA:ADC FVH:STA FVH
4180 LDA#3:JMP SR
4190
4200.dma
4210 LDY#1:LDA (PL),Y:STA R1L
4220 INY:LDA (PL),Y:STA R1H
4230 DEY:LDA (R1L),Y:DEY:ORA (R1L),Y
4240 BEQ DMY:LDA#33:JMP EXIT:.DMY
4250 INY:LDA FVH:STA (R1L),Y:STA R5H
4260 DEY:LDA FVL:STA (R1L),Y:STA R5L
4270 PLA:STA R4L:STA (R5L),Y
4280 INY:PLA:STA R4H:STA (R5L),Y
4290 LDA R4L:SEC:SBC#1
4300 CLC:ROL A:CLC:ADC#4
4310 CLC:ADC R5L:STA R6L
4320 LDA#0:ADC R5H:STA R6H
4330 INY:LDA R6L:STA (R5L),Y
4340 INY:LDA R6H:STA (R5L),Y
4350 STY R7H:LDA#0:STA R7L:STA R2H
4360 LDA#2:STA R2L
4370.DMLB LDA#0:STA R3L:STA R3H
4380 PLA:CLC:ADC#1:STA R1L
4390 PLA:ADC#0:STA R1H
4400 JSR AX:INC R7L
4410 LDA R7L:CMP R4L:BEQ DMLA
4420 LDY R7H
4430 INY:LDA R3L:STA (R5L),Y:STA R2L
4440 INY:LDA R3H:STA (R5L),Y:STA R2H
4450 LDA#0:STA R3L:STA R3H:BEQ DMLB
4460.DMLA
4470 LDA R6L:CLC:ADC R3L:STA FVL
4480 LDA R6H:ADC R3H:STA FVH
4490 LDY#0
4500.DAL LDA#0:STA (R6L),Y
4510 LDA FVH:CMP R6H:BNE DMI
4520 LDA FVL:CMP R6L:BEQ DMX
4530.DMI INCR6L:BNE DAL:INCR6H:BNE DAL
4540.DMX LDA#3:JMP SR
4550.GTA
4560 PLA:STA R7L:PLA:STA R7H
4570 LDY#1:LDA (PL),Y:STA R1L
4580 INY:LDA (PL),Y:STA R1H
4590 DEY:LDA (R1L),Y:STA R5H
4600 DEY:LDA (R1L),Y:STA R5L
4610 ORA R5H:BEQ GTZ
4620 LDA (R5L),Y:STA R4L
4630 LDY#2:LDA (R5L),Y:STA R3L
4640 INY:LDA (R5L),Y:STA R3H
4650 PLA:STA R1L:PLA:STA R1H
4660 CLC:ROL R1L:ROL R1H
4670 LDA R1L:CLC:ADC R3L:STA R3L
4680 LDA R1H:ADC R3H:STA R3H
4690 LDA R4L:CMP#1:BEQ GTX
4700 LDA#1:STA R4H
4710.GTL PLA:STA R1L:PLA:STA R1H
4720 INY:LDA (R5L),Y:STA R2L
4730 INY:LDA (R5L),Y:STA R2H
4740 JSR AX:INC R4H
4750 LDA R4H:CMP R4L:BNE GTL
4760.GTX LDA R7H:PHA:LDA R7L:PHA:RTS
4770.GTZ LDA#11:JMP EXIT
4780.AX
4790 LDX#8:LDA R1H:BEQ DML
4800 LDY R2H:STA R2H:STY R1H
4810 LDA R1L:LDY R2L:STA R2L:STY R1L
4820.DML CLC:ROR R2H:ROR R2L:BCS DMA
4830.DMC CLC:ROL R1L:ROL R1H
4840 DEX:BNE DML:RTS
4850.DMA LDA R1L:CLC:ADC R3L:STA R3L
4860 LDA R1H:ADC R3H:STA R3H:JMP DMC
4870
4880.vds
4890 LDY#1:LDA (PL),Y
4900 TAX:INY
4910.DS LDA(PL),Y:JSR OSWRITE
4920 INY:DEX:BNE DS
4930 TYA:JMP SR
4940
4950.cli
4960 LDA PL:CLC:ADC#2:TAX
4970 LDA PH:ADC#0:TAY
4980 JSR OSCLI
4990 LDY#1:LDA (PL),Y:CLC:ADC#2
5000 JMP SR
5010
5020.sqr
5030 PLA:STA R4L:STA R5L
5040 PLA:STA R4H:STA R5H
5050 BMI QE
5060 CLC:ROR R5H:ROR R5L:LDA#10:STA SG
5070.QA
5080 LDA R4L:STA R2L:LDA R4H:STA R2H
5090 LDA R5L:STA R3L:LDA R5H:STA R3H
5100 JSR DM
5110 LDA R2L:CLC:ADC R5L:STA R5L
5120 LDA R2H:ADC R5H:STA R5H
5130 CLC:ROR R5H:ROR R5L
5140 DEC SG:BNE QA
5150 LDA R5H:PHA:LDA R5L:PHA
5160 JMP SX
5170.QE LDA#22:JMP EXIT
5180
5190.sgn
5200 PLA:STA R1L:PLA:BMI SM
5210 ORA R1L:BEQ SZ
5220 LDA#0:PHA:LDA#1:PHA:JMP SR
5230.SM LDA#&FF:PHA:PHA:JMP SX
5240.SZ LDA#0:PHA:PHA:JMP SX
5250
5260.abs
5270 PLA:STA R1L:PLA:BMI AM
5280 PHA:LDA R1L:PHA:JMP SX
5290.AM STA R1H:LDA#0:SEC:SBC R1L
5300 TAX:LDA#0:SBC R1H:PHA:TXA:PHA
5310 JMP SX
5320
5330.get LDA#0:PHA
5340.gp JSR OSRDCH:BCS ge
5350 PHA:JMP SX
5360.ge CMP#27:BNE gp
5370 LDA#126:JSR OSBYTE:BNE gp
5380
5390.adv
5400 PLA:TAX:PLA:TAY:LDA#&80
5410 JSR OSBYTE:TYA:PHA:TXA:PHA
5420 JMP SX
5430
5440.pos
5450 LDA#&86:JSR OSBYTE:LDA#0:PHA
5460 TXA:PHA:JMP SX
5470
5480.vpo
5490 LDA#&86:JSR OSBYTE:LDA#0:PHA
5500 TYA:PHA:JMP SX
5510
5520.poi
5530 PLA:STA R2L:PLA:STA R2H
5540 PLA:STA R1L:PLA:STA R1H
5550 LDX#R1L:LDY#0:LDA#&9:JSR OSWORD
5560 LDA R3L:CMP#&FF:BEQ PM
5570 LDA#0:PHA:LDA R3L:PHA
5580 JMP SX
5590.PM PHA:PHA:LDA#1:JMP SR
5600
5610.cal
5620 PLA:STA R1L:PLA:STA R1H
5630 LDA#((CR-1)DIV256):PHA
5640 LDA#((CR-1)MOD256):PHA
5650 LDA&40C:ROL A:LDX&460:LDY&464
5660 LDA&404:JMP(R1L)
5670.CR JMP SX
5680
5690.not
5700 PLA:STA R1L:PLA:STA R1H
5710 LDA#&FF:SEC:SBC R1L:TAY
5720 LDA#&FF:SBC R1H:PHA:TYA:PHA
5730 JMP SX
5740
5750.env
5760 LDX#13:LDY#14
5770.EL PLA:STA R1L,X:PLA
5780 DEX:DEY:BNE EL
5790 LDX#R1L:LDY#0:LDA#&8:JSR OSWORD
5800 JMP SX
5810
5820.rtf
5830 PLA:TAY:PLA:TAX
5840 PLA:STA PL:PLA:STA PH
5850 TXA:PHA:TYA:PHA
5860 JMP SF
5870
5880.nxt
5890 LDY#0:LDA FSL:CLC:ADC (FAL),Y
5900 STA (FAL),Y:STA R3L
5910 INY:LDA FSH:ADC (FAL),Y
5920 STA (FAL),Y:STA R3H
5930 LDA FSH:BPL NP
5940 LDA R3L:SEC:SBC FLL
5950 LDA R3H:SBC FLH
5960 BVS NN:BMI NE:BPL NN
5970.NP LDA FLL:SEC:SBC R3L
5980 LDA FLH:SBC R3H:BVS NN:BMI NE
5990.NN LDA FJL:STA PL:LDA FJH:STA PH
6000 JMP SF
6010.NE LDX FP:LDY#0:DEX
6020.NXL LDA &500,X:STA FAL,Y
6030 DEX:INY:CPY#8:BNE NXL:INX:STX FP
6040 JMP SX
6050
6060.EXIT
6070 STA SG
6080 LDA SP:TSX:STX SP:TAX:TXS
6090 LDA RH:PHA:LDA RL:PHA
6100 RTS
6110]
6120NEXT
6130ENDPROC
6140
6150DEFPROCSETUP
6160PL=&70:PH=&71:JTL=&72:JTH=&73
6170R1L=&74:R1H=&75:R2L=&76:R2H=&77
6180R3L=&78:R3H=&79:R4L=&7A:R4H=&7B
6190R5L=&7C:R5H=&7D:R6L=&7E:R6H=&7F
6200R7L=&80:R7H=&81
6210SP=&82:SG=&83:RL=&84:RH=&85
6220FAL=&86:FAH=&87:FLL=&88:FLH=&89
6230FSL=&8A:FSH=&8B:FJL=&8C:FJH=&8D
6240FP=&8E
6250OSWRITE=&FFEE:OSBYTE=&FFF4
6260OSWORD=&FFF1:OSCLI=&FFF7
6270OSRDCH=&FFE0
6280P%=page%
6290PROCCOPY:JT=P%
6300RT=JT+160
6310ENDPROC
6320
6330DEFPROCJUMPTAB
6340PROCE(add,2):PROCE(sub,4)
6350PROCE(mul,6):PROCE(div,8)
6360PROCE(gtt,10):PROCE(ltt,12)
6370PROCE(geq,14):PROCE(leq,16)
6380PROCE(equ,18):PROCE(neq,20)
6390PROCE(neg,22):PROCE(vdu,24)
6400PROCE(ink,26):PROCE(rts,28)
6410PROCE(ldi,30):PROCE(sti,32)
6420PROCE(end,34):PROCE(pri,36)
6430PROCE(dup,38):PROCE(swp,40)
6440PROCE(drp,42):PROCE(mod,44)
6450PROCE(vdd,46):PROCE(vdl,48)
6460PROCE(snd,50):PROCE(ldt,52)
6470PROCE(stt,54):PROCE(and,56)
6480PROCE(ora,58):PROCE(eor,60)
6490PROCE(rnd,62):PROCE(jmp,64)
6500PROCE(jsr,66):PROCE(jmc,68)
6510PROCE(lda,70):PROCE(sta,72)
6520PROCE(ldl,74):PROCE(fxr,76)
6530PROCE(ldm,80):PROCE(stm,82)
6540PROCE(dmm,84):PROCE(dma,86)
6550PROCE(vds,90):PROCE(cli,92)
6560PROCE(sqr,102):PROCE(sgn,104)
6570PROCE(abs,106):PROCE(get,108)
6580PROCE(adv,110):PROCE(pos,112)
6590PROCE(vpo,114):PROCE(poi,116)
6600PROCE(cal,118):PROCE(not,120)
6610PROCE(env,122)
6620PROCE(rtf,126):PROCE(nxt,132)
6630ENDPROC
6640
6650DEFPROCE(opr,num)
6660JT?num=opr MOD256
6670JT?(num+1)=opr DIV256
6680ENDPROC
6690
6700DEFPROCGTEST
6710P%?4=34:!P%=(P%+5)+(P%+5)*&10000
6720P%=P%+5
6730ENDPROC
6740
6750DEFPROCCOPY
6760CC%=TOP
6770REPEAT CC%=CC%-1:UNTIL ?CC%=&FA
6780IF CC%?1<>58 ORCC%?2<>&FA GOTO6770
6790CC%=CC%+3
6800FOR CX%=CC%TO TOP
6810?P%=?CX%:P%=P%+1:NEXT:P%=P%-1
6820SLOC%=P%:P%=P%+100
6830ENDPROC
6840
6850DEFPROCFINISH
6860toprts%=P%:!(SLOC%+4)=toprts%
6870PROCGTEST
6880$(SLOC%+8)="*SAVE RTS "+STR$~
 page%+" "+STR$~(P%+1 )
6890PRINT''"<<< TOP OF RTS IS =
 ";toprts%;" >>>"
6900PRINT'"This number should be entered
 in the"'"compiler."''"Reseting PAGE to
 its usual value"'"and then typing RUN will
 allow you"'"to save your run time syst
 em."''
6910ENDPROC
6920
6930REM type in line 6960 exactly as
6940REM shown.
6950
6960STOP:STOP
6970REM
6980REM ================================
6990REM  T.C. BASIC RTS
7000REM  (c) D.J. Pilling
7010REM  January 1985
7020REM ================================
7030MODE7:PRINT"T.C. BASIC RTS "
7040LOMEM=TOP+40:HIMEM=TOP+100
7050T%=TOP:O%=T%+8:L%=O%+6:S%=T%!4
7060PRINT"MODULE CONTAINS "$L%
7070X%=O%MOD256:Y%=O%DIV256
7080PRINT"(1) LOAD NEW CODE "
7090PRINT"(2) RUN  CODE "
7100PRINT"(3) SAVE MODULE"
7110I%=GET:I%=I%-ASC"0":IFI%<1 OR
  I%>3 GOTO7110
7120ONI%GOTO 7130,7220,7270
7130INPUT"FILE NAME "$L%:*OPT1,1
7140*OPT2,1
7150H%=OPENUP$L%
7160INPUT#H%,C%,F%,E%,$L%
7170FOR J%=0 TO E%-C%+3:S%?J%=BGET#H%:NEXT
7180CLOSE#H%
7190*OPT
7200$L%="M."+RIGHT$($L%,LEN$L%-2)+"
  "+STR$~PAGE+" "+S TR$~(J%+S%+1)
7210GOTO7030
7220?&70=S%MOD256:?&71=S%DIV256
7230CALL (T%+260)
7240PRINT'':IF?&83=44PRINT"END OF CODE
  "ELSEIF?&83=88PRINT"DIVISION BY ZERO
  "ELSEIF?&83=22PRINT"NEGATIVE SQUARE
  ROOT"ELSEIF?&83=11PRINT"NO ARRAY DIM
  "ELSEIF?&83=33PRINT"BAD DIM"
7250PRINT"<RETURN> to reset ";
7260REPEAT UNTIL GET=13:GOTO7030
7270CALL&FFF7:GOTO7030
```

■

```
6710P%?4=34:!P%=(P%+5)+(P%+5)*&10000
6720P%=P%+5
6730ENDPROC
6740
6750DEFPROCCOPY
6760CC%=TOP
6770REPEAT CC%=CC%-1:UNTIL ?CC%=&FA
6780IF CC%?1<>58 ORCC%?2<>&FA GOTO6770
6790CC%=CC%+3
6800FOR CX%=CC%TO TOP
6810?P%=?CX%:P%=P%+1:NEXT:P%=P%-1
6820SLOC%=P%:P%=P%+100
6830ENDPROC
6840
6850DEFPROCFINISH
6860toprts%=P%:!(SLOC%+4)=toprts%
6870PROCGTEST
6880$(SLOC%+8)="*SAVE RTS "+STR$~
 page%+" "+STR$~(P%+1 )
6890PRINT''"<<< TOP OF RTS IS =
 ";toprts%;" >>>"
6900PRINT'"This number should be entered
 in the"'"compiler."''"Reseting PAGE to
 its usual value"'"and then typing RUN will
 allow you"'"to save your run time syst
 em."''
6910ENDPROC
6920
6930REM type in line 6960 exactly as
6940REM shown.
6950
6960STOP:STOP
6970REM
6980REM ==============================
6990REM  T.C. BASIC RTS
7000REM  (c) D.J. Pilling
7010REM  January 1985
7020REM ==============================
7030MODE7:PRINT"T.C. BASIC RTS "
7040LOMEM=TOP+40:HIMEM=TOP+100
7050T%=TOP:O%=T%+8:L%=O%+6:S%=T%!4
7060PRINT"MODULE CONTAINS "$L%
7070X%=O%MOD256:Y%=O%DIV256
7080PRINT"(1) LOAD NEW CODE "
7090PRINT"(2) RUN  CODE "
7100PRINT"(3) SAVE MODULE"
7110I%=GET:I%=I%-ASC"0":IFI%<1 OR
  I%>3 GOTO7110
7120ONI%GOTO 7130,7220,7270
7130INPUT"FILE NAME "$L%:*OPT1,1
7140*OPT2,1
7150H%=OPENUP$L%
7160INPUT#H%,C%,F%,E%,$L%
7170FOR J%=0 TO E%-C%+3:S%?J%=BGET#H%:NEXT
7180CLOSE#H%
7190*OPT
7200$L%="M."+RIGHT$($L%,LEN$L%-2)+"
  "+STR$~PAGE+" "+S TR$~(J%+S%+1)
7210GOTO7030
7220?&70=S%MOD256:?&71=S%DIV256
7230CALL (T%+260)
7240PRINT'':IF?&83=44PRINT"END OF CODE
 "ELSEIF?&83=88PRINT"DIVISION BY ZERO
 "ELSEIF?&83=22PRINT"NEGATIVE SQUARE
 ROOT"ELSEIF?&83=11PRINT"NO ARRAY DIM
 "ELSEIF?&83=33PRINT"BAD DIM"
7250PRINT"<RETURN> to reset ";
7260REPEAT UNTIL GET=13:GOTO7030
7270CALL&FFF7:GOTO7030
```

■